SSN COLLEGE OF ENGINEERING, KALAVAKKAM

(An Autonomous Institution, Affiliated to Anna University, Chennai)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

UCS2611 – INTERNET PROGRAMMING LAB

Team:

Ashwini Parvatha G S 3122215001016

A Bhavana 3122215001019

Charumathi P 3122215001020

Problem Statement:

Develop a full stack web application for conducting On-line quiz using MVC architecture. The application should facilitate the normal and admin users to access it. To the normal user, instructions, questions with options and the score needs to be provided as the following snapshots. Admin user can view the registered users and their scores.

Design the following:

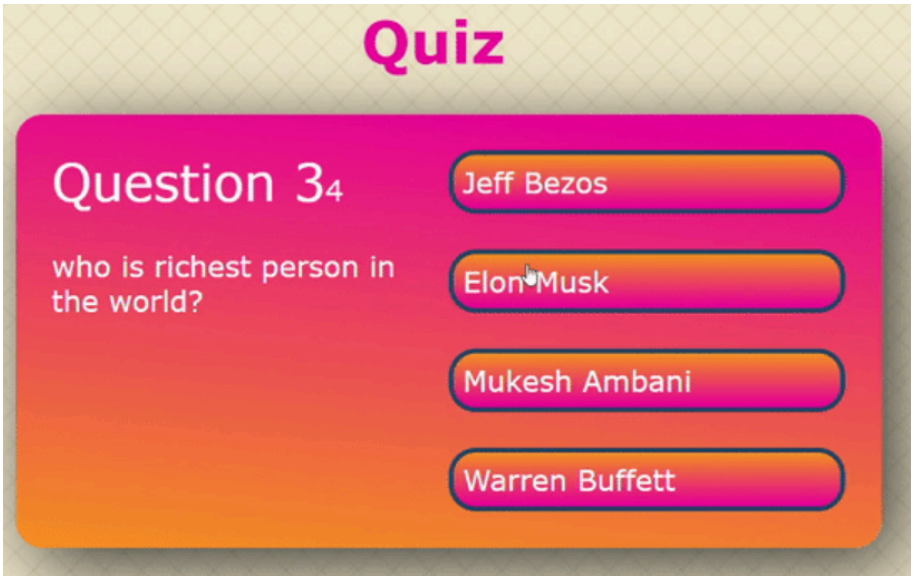
- Schema of the necessary MongoDB collections. One of which is shown below
  - Create Questions collection in MongoDB as follows to store the information about the students and Questions with choices.

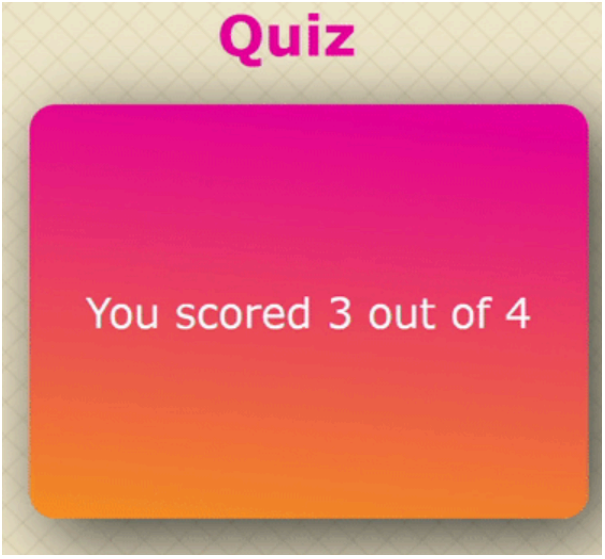
Question	Choice_ A	Choice_ B	Choice_ C	Choice_ D	Answer

- Design the application with the necessary endpoints, controllers, collections, and components as a sequence diagram

Do the following operations:

- Authenticate the user and provide necessary messages
- Keep a timer for the Quiz
- Sending appropriate GET http requests from front end, to the endpoint in the node server.
- Perform necessary operations in the Mongo Collection at the endpoints
- Create a suitable interface in ReactJS to display the results

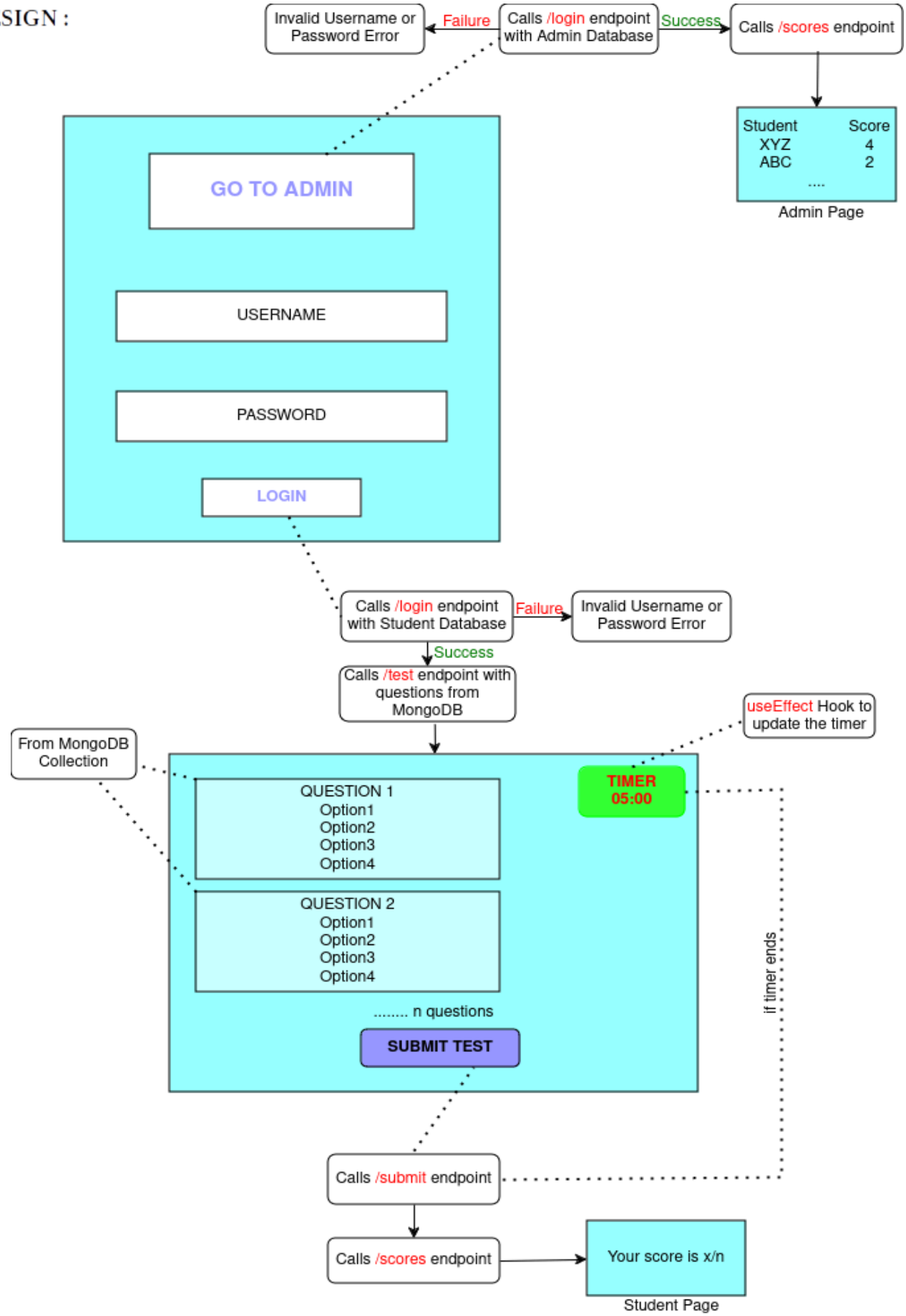




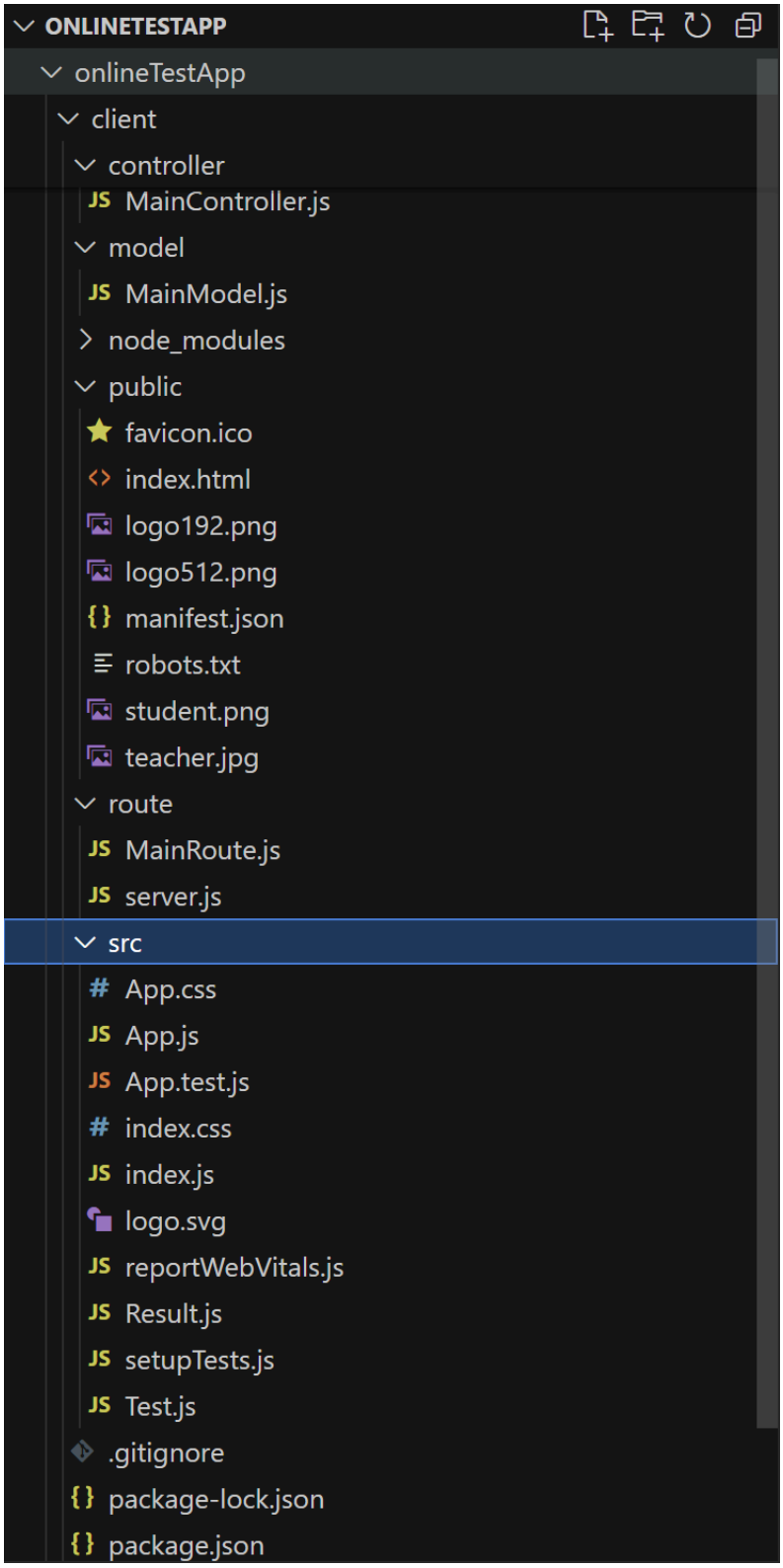
Use the following Best Practices:

- Design before coding
- Incremental coding
- Usage of proper naming convention
- Usage of Comments to the code
- Indentation of code

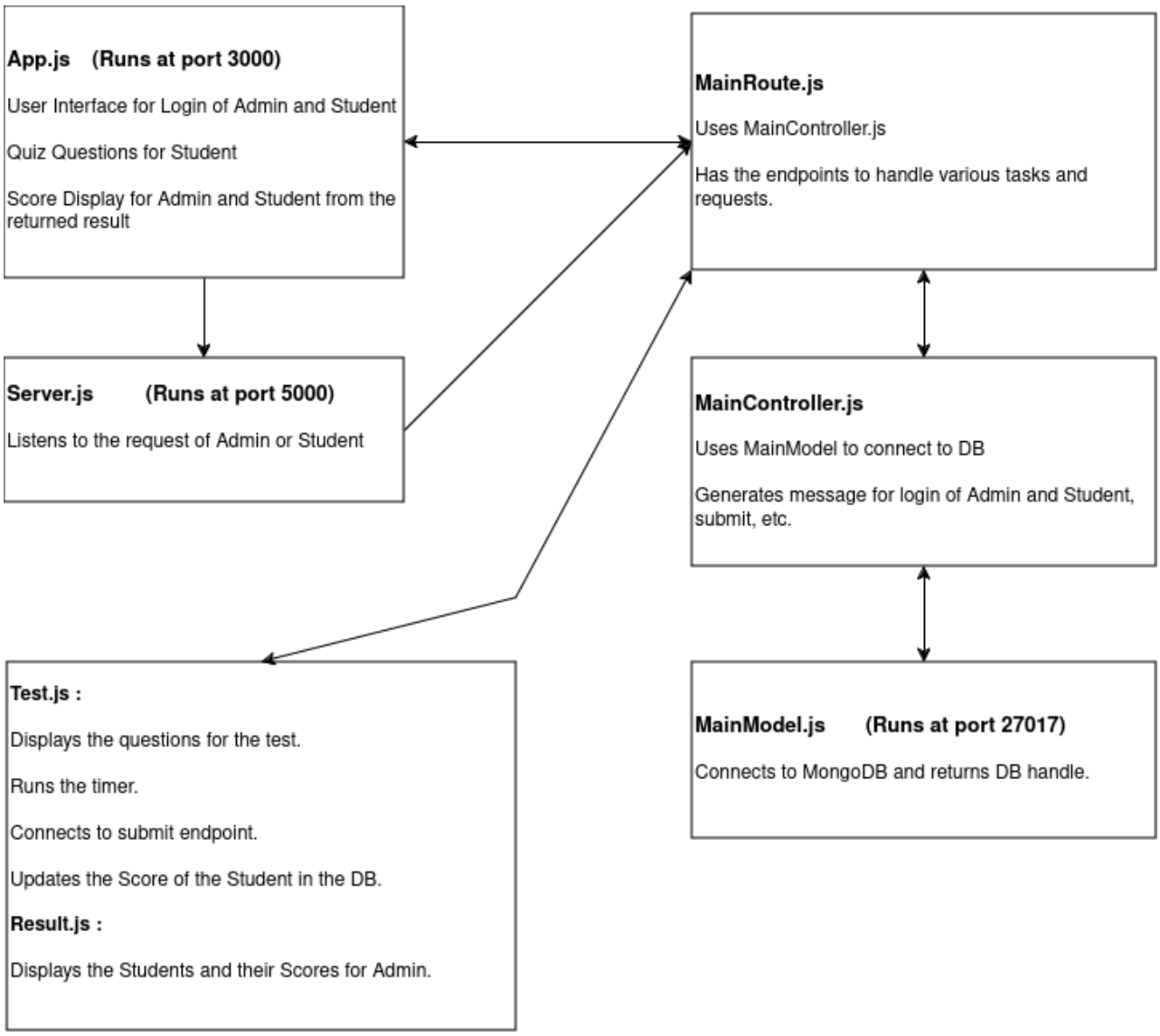
DESIGN :


Invalid Username or Password Error
Failure
Calls /login endpoint with Admin Database
Success
Calls /scores endpoint
Student Score
XYZ 4
ABC 2
....
Admin Page
GO TO ADMIN
USERNAME
PASSWORD
LOGIN
Calls /login endpoint with Student Database
Failure
Invalid Username or Password Error
Success
Calls /test endpoint with questions from MongoDB
useEffect Hook to update the timer
From MongoDB Collection
TIMER 05:00
QUESTION 1
Option1
Option2
Option3
Option4
QUESTION 2
Option1
Option2
Option3
Option4
..... n questions
SUBMIT TEST
if timer ends
Calls /submit endpoint
Calls /scores endpoint
Your score is x/n
Student Page

Directory Structure:



MODEL VIEW CONTROLLER (MVC) EXECUTION FLOW :



MongoDB Schemas:

students collection :

- username - text
- password - text

admin collection :

- username - text
- password - text

questions collection :

- question - text
- options- array of answer options
- correct_ans - text

results collection :

- username - text
- score - numeric
- time - string

```
test> use ipLab
switched to db ipLab
ipLab> db.students.findOne()
{
  _id: ObjectId("6633d27744a872598a49305c"),
  username: 'charu',
  password: '123'
}
ipLab> db.admin.findOne()
{
  _id: ObjectId("6633d28b44a872598a49305d"),
  username: 'admin1',
  password: '123'
}
ipLab> db.questions.findOne()
{
  _id: ObjectId("66335b458366513ab384e9ec"),
  question: 'What is the capital of France?',
  options: [ 'Paris', 'Berlin', 'London', 'Rome' ],
  correct_answer: 'Paris'
}
ipLab> db.results.findOne()
{
  _id: ObjectId("6634ab36ac2af03edc59b84d"),
  name: 'Charumathi',
  score: 4,
  time: '04:06'
}
ipLab>
```

Code:

App.js

```
import React, { useState } from 'react';
import Test from './Test';
import Result from './Result';
import './App.css';

function App() {
  const [message, setMessage] = useState("");
  const [isLoggedIn, setLogIn] = useState(false);
  const [user, setUser] = useState("");
  const [showAdminPanel, setShowAdminPanel] = useState(false);
  const sendPOSTmethod = () => {
    var userName = document.getElementById('username').value;
    var Password = document.getElementById('password').value;
    const mode = document.getElementById('mode').value;
    fetch('http://localhost:5000/login', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({ username: userName, password: Password, mode:
mode }),
    })
    .then(res => res.json())
    .then(data => {
      setMessage(data.message);
      console.log(data.message);
      setUser(data.message);
      if (data.message !== "incorrect username or password") {
        setLogIn(true);
      }
    })    .catch(error => console.error('Error:', error));
  };
  const toggleAdminPanel = () => {
    setShowAdminPanel(!showAdminPanel);
  };
  return (
    <div className="container p-5 my-5 bg-white rounded">

      {showAdminPanel ?(<div>
      {isLoggedIn ? (<Result />):
    (<div className="sidebyside"><div className="insidebyside"><center>
      <button className="btn btn-secondary mb-3"
onClick={toggleAdminPanel}>
                          {showAdminPanel ? 'Go To Student' : 'Go To
Admin'}
      </button><br/>
      <label > Username <input class="form-control" type="text"
id='username' /></label>
      <br/>
```

```
      <label> Password <input class="form-control" type="text"
id='password'/></label>
      <br/>
      <input type="hidden" id="mode" value="admin"></input>
      <button className="btn btn-primary mb-3"
onClick={sendPOSTmethod}>Login</button>
      <p>{typeof message === 'object' ? JSON.stringify(message) :
message}</p></center>
      </div><div className="insidebyside"><img src="./teacher.jpg"
style={{width:'300px'}}></img></div></div>) }</div>
      ):(<div>
        {isLoggedIn ? (< Test name={user}/>):
       (<div className="sidebyside"><div className="insidebyside"><center>
        <button className="btn btn-secondary mb-3"
onClick={toggleAdminPanel}>
                          {showAdminPanel ? 'Go To Student' : 'Go To
Admin'}
      </button><br/>
        <label > Username <input class="form-control" type="text"
id='username' /></label>
        <br/>
        <label> Password <input class="form-control" type="text"
id='password'/></label>
        <br/>
        <input type="hidden" id="mode" value="student"></input>
        <button className="btn btn-primary mb-3"
onClick={sendPOSTmethod}>Login</button>
        <p>{typeof message === 'object' ? JSON.stringify(message) :
message}</p></center>
        </div><div className="insidebyside"><img src="./student.png"
style={{width:'300px'}}></img></div></div>) }</div>)}</div>
  );
}
export default App;
```

Components

Test.js

```
import React, { useState, useEffect } from 'react';
import './App.css';

function Test(name) {
  const [questions, setQuestions] = useState([]);
  const [score, setScore] = useState(0);
  const [submitted, setSubmitted] = useState(false);
  const [seconds, setSeconds] = useState("59");
  const [minutes, setMinutes] = useState("04");

  const countDown = () => {
    var curr_mins = parseInt(minutes);
    var curr_secs = parseInt(seconds);
```

```
  if (curr_secs == 1){
    curr_secs+=60;
  }
  curr_secs = curr_secs - 1;
  setSeconds(String(curr_secs));
  if (curr_secs == 60){
    curr_mins = curr_mins - 1;
    setMinutes("0"+String(curr_mins));
  }
  if (curr_secs < 10){
    setSeconds("0"+String(curr_secs));
  }
  if (curr_mins == "00"){
    document.getElementById("timer").style.background = "red";
  }
  if (parseInt(curr_mins) < 0){
    handleSubmit();
  }
}
useEffect(() => {
  fetchQuestions();
}, []);

useEffect(() => {if(!submitted){setTimeout(countDown, 1000)}});

const fetchQuestions = () => {
  fetch('http://localhost:5000/test', {
    method: 'GET',
    headers: {
      'Content-Type': 'application/json',
    },
  })
  .then(res => res.json())
  .then(data => {
    const updatedQuestions = data.map(question => ({
      ...question,
      selectedOption: null // Adding selectedOption property
    }));
    setQuestions(updatedQuestions);
  })
  .catch(error => console.error('Error:', error));
};
const writeResults = (userResult) => {
  fetch('http://localhost:5000/submit', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify(userResult),
  })
  .then(res => res.json())
  .then(data => { console.log(data);})
  .catch(error => console.error('Error:', error));
};
```

```
  const handleOptionChange = (questionId, selectedOption) => {
    const updatedQuestions = questions.map(question => {
      if (question._id === questionId) {
        return {
          ...question,
          selectedOption: selectedOption
        };
      }
      return question;
    });
    setQuestions(updatedQuestions);
  };

  const handleSubmit = () => {
    // Calculate score
    document.getElementById("timer-box").style.visibility = "hidden";
    let userScore = 0;
    questions.forEach(question => {
        console.log(question.selectedOption,question.correct_answer)
      if (question.selectedOption === question.correct_answer) {
        userScore++;
      }
    });
    var secs = String(60-seconds);
    var mins = String(4-minutes);
    setScore(userScore);
    if ((60-seconds)<10){
      secs = "0" + String(60-seconds);
    }
    if ((2-minutes)<10){
      mins = "0" + String(4-minutes);
    }
    // Create the object with user details and score
    const userResult = {
      name: name.name,
      score: userScore,
      time: mins + ":" + secs // Set time as 0 for now
    };
    console.log(userResult);
    writeResults(userResult);
    // Set submitted flag to true
    setSubmitted(true);
  };

  return (

    <div>
              <div className="sticky" id="timer-box" ><h1 className="timer"
id="timer">{String(minutes)+":"+String(seconds)}</h1></div>
        <form className='form'>
      {!submitted ? (
        <div>
          {questions.map(question => (
            <div key={question._id}>
```

```
              <div><b>{question.question}</b></div>
              <ul>
                {question.options.map((option, index) => (
                  <li key={index}>
                    <label class="form-label">
                      <input
                        class="form-check-input"
                        type="radio"
                        name={question._id}
                        value={option}
                        onChange={() => handleOptionChange(question._id,
option)}
                      />
                      {option}
                    </label>
                  </li>
                ))}
              </ul>
            </div>
          ))}
          <button className="btn btn-success mb-3"
onClick={handleSubmit}>Submit</button>
        </div>
      ) : (
        <div>
          <h2>Thank you for submitting!</h2>
          <p>Your score: {score}</p>
        </div>
      )}
      </form>
    </div>
  );
}

export default Test;
```

Result.js

```
import React, { useState, useEffect } from 'react';
import './App.css';

function Results() {
  const [scores, setScores] = useState([]);

  useEffect(() => {
    fetchResults();
  }, []);

  const fetchResults = () => {
    fetch('http://localhost:5000/scores', {
      method: 'GET',
      headers: {
        'Content-Type': 'application/json',
      },
```

```
    })
    .then(res => res.json())
    .then(data => {setScores(data)
    })
    .catch(error => console.error('Error:', error));
  };
  return (
    <div className="container">
      <h2>Results</h2>
      <table class="table">
        <thead>
          <tr>
            <th>Name</th>
            <th>Score</th>
            <th>Time</th>
          </tr>
        </thead>
        <tbody>
          {scores.map((result, index) => (
            <tr key={index}>
              <td>{result.name}</td>
              <td>{result.score}</td>
              <td>{result.time}</td>
            </tr>
          ))}
        </tbody>
      </table>
    </div>
  );
}

export default Results;
```

server.js

```
const express = require('express');
const bodyParser = require('body-parser');
const cors = require('cors');
const mainRoute = require('./MainRoute');

const app = express();
const PORT = process.env.PORT || 5000;

app.use(bodyParser.json());
app.use(cors());

app.use('/', mainRoute);


app.listen(PORT, () => {
    console.log("Server Listening on port", PORT);
});
```

MainRoute.js

```
const { generateMessage, getQuestions, addResults, getResults } =
require('../controller/MainController');

const express = require('express');
const router = express.Router();

router.post("/login", async function (req, res) { // Make the route
handler asynchronous
    const { username, password,mode } = req.body;
    try {
        const data = await generateMessage(username, password,mode); //
Wait for the Promise to resolve
        console.log("DEBUG ", data);
        res.json({ message: data });
    } catch (error) {
        console.error("Error generating message:", error);
        res.status(500).json({ message: 'Internal server error' });
    }
});
router.get("/test", async function (req, res) { // Make the route handler
asynchronous
    try {
        const data = await getQuestions(); // Wait for the Promise to
resolve
        console.log("DEBUG ", data);
        res.json(data );
    } catch (error) {
        console.error("Error generating message:", error);
        res.status(500).json('Internal server error' );
    }
});
router.post("/submit", async function (req, res) { // Make the route
handler asynchronous
    try {
        const curr = req.body;
        console.log(curr);
        const data = await addResults(curr); // Wait for the Promise to
resolve
        res.json(data );
    } catch (error) {
        console.error("Error generating message:", error);
        res.status(500).json('Internal server error' );
    }
});
router.get("/scores", async function (req, res) { // Make the route
handler asynchronous
    try {
        const data = await getResults(); // Wait for the Promise to
resolve
        console.log("DEBUG ", data);
        res.json(data );
    } catch (error) {
```

```
        console.error("Error generating message:", error);
        res.status(500).json('Internal server error' );
    }
});
module.exports = router;
```

MainController.js

```
const { connect } = require('../model/MainModel');

async function generateMessage(username, password, mode) {
    console.log("I am generateMessage Function...");
    console.log("Username:", username);
    console.log("Password:", password);
    var table = "";
    if (mode !== "admin"){
        table = 'students'
    }
    else{
        table = 'admin'
    }
    try {
        // Connect to MongoDB
        const db = await connect();
        console.log("DEBUG DB... 1");
        // Define collection
        const studentCollection = db.collection(table);
        console.log("DEBUG DB... 2");
        // Find user by username
        const student = await studentCollection.findOne({ username });
        console.log("DEBUG DB... 3");
        if (!student) {
            return 'incorrect username or password'; // Return empty
strings if user not found
        }
        console.log("DEBUG DB... 4");
        // Check password
        if (student.password !== password) {
            return 'incorrect username or password'; // Return empty
strings if password is incorrect
        }
        console.log("Success. ", student.username + " " +
student.password);
        return student.username;
    } catch (error) {
        console.log("Error logging in:", error);
        throw error;
    }
}
async function getQuestions() {
        try {
            // Connect to MongoDB
            const db = await connect();
            const ques = db.collection("questions");
```

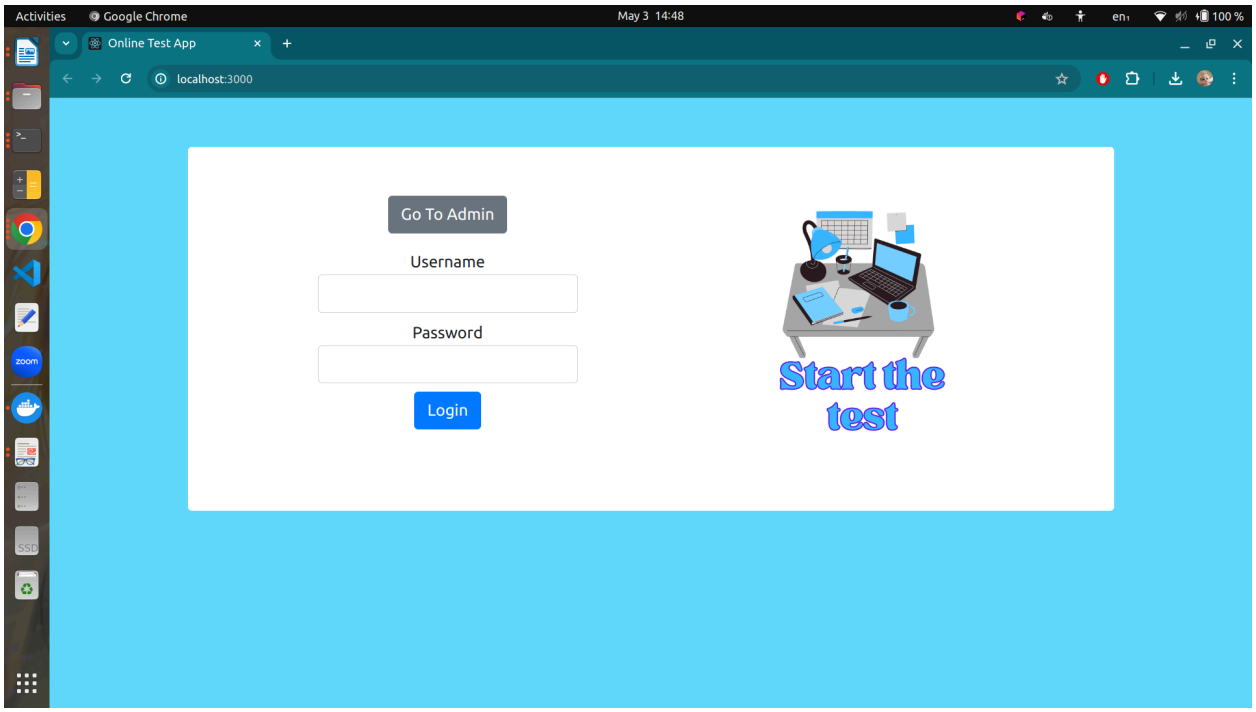
```
            const questions = await ques.find({}).toArray();
            return questions;
        } catch (error) {
            console.log("Error logging in:", error);
            throw error;
        }
    }
async function addResults(currRes) {
            try {
                // Connect to MongoDB
                const db = await connect();
                const res = db.collection("results");
                const mes = await res.insertOne(currRes);
                return mes;
            } catch (error) {
                console.log("Error logging in:", error);
                throw error;
            }
        }
async function getResults(currRes) {
                try {
                    // Connect to MongoDB
                    const db = await connect();
                    const res = db.collection("results");
                     const results = await res.find({}).toArray();
                    return results;
                } catch (error) {
                    console.log("Error logging in:", error);
                    throw error;
                }
            }
module.exports = { generateMessage, getQuestions, addResults, getResults
};
```

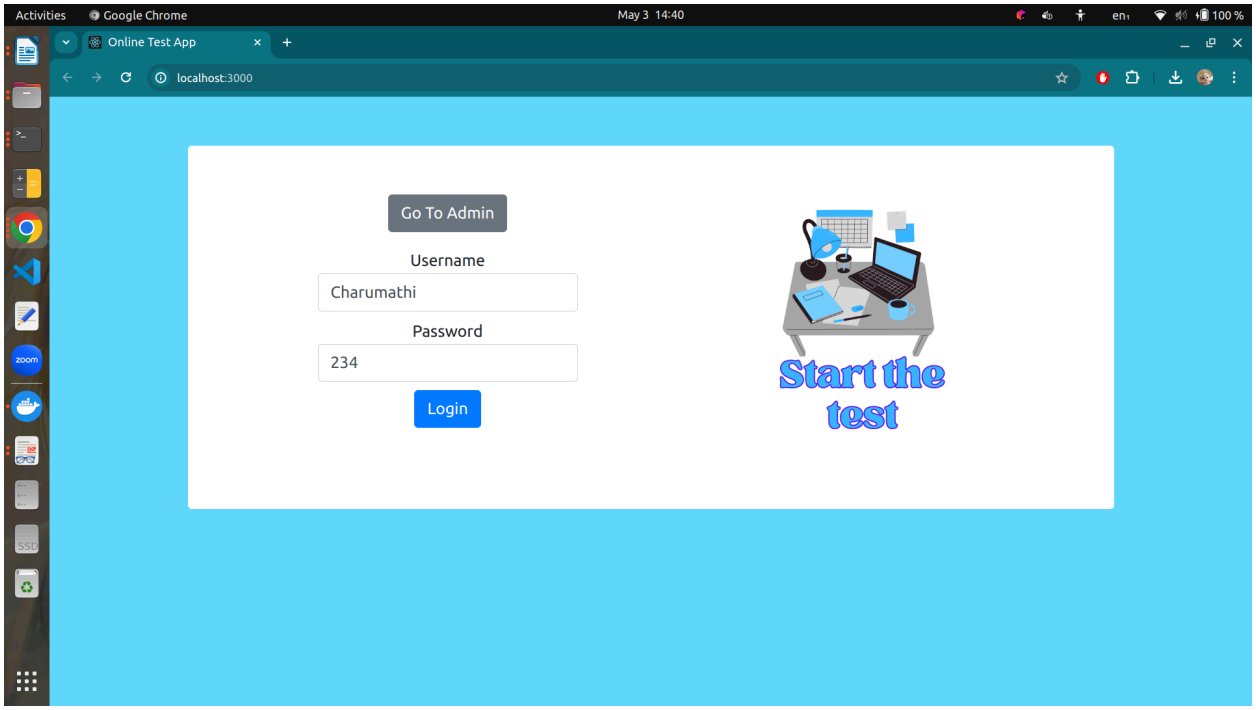
MainModel.js

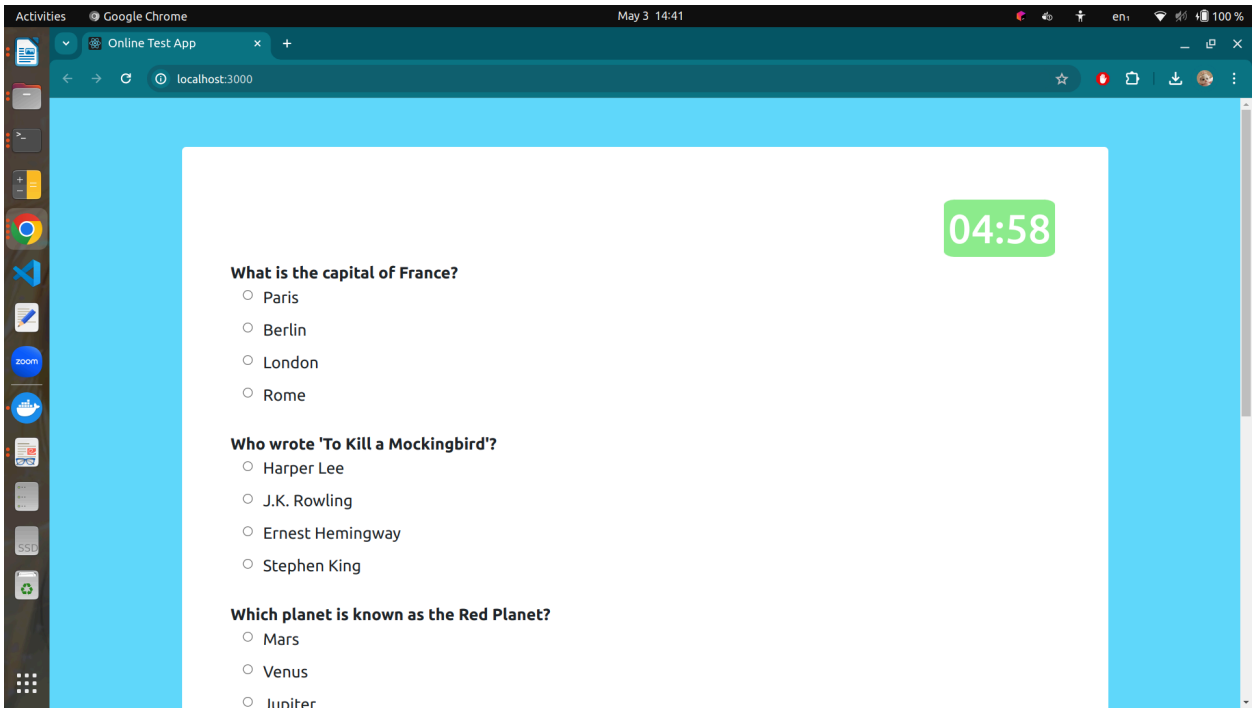
```
const { MongoClient } = require('mongodb');

const uri = "mongodb://localhost:27017";
const dbName = "ipLab";

// Function to connect to MongoDB
async function connect() {
  try {
    // Create a new MongoClient instance
    const client = new MongoClient(uri);

    // Connect to MongoDB
    await client.connect();
    console.log("Connected to MongoDB server");

    // Access the database
    const db = client.db(dbName);
    console.log('out');
```

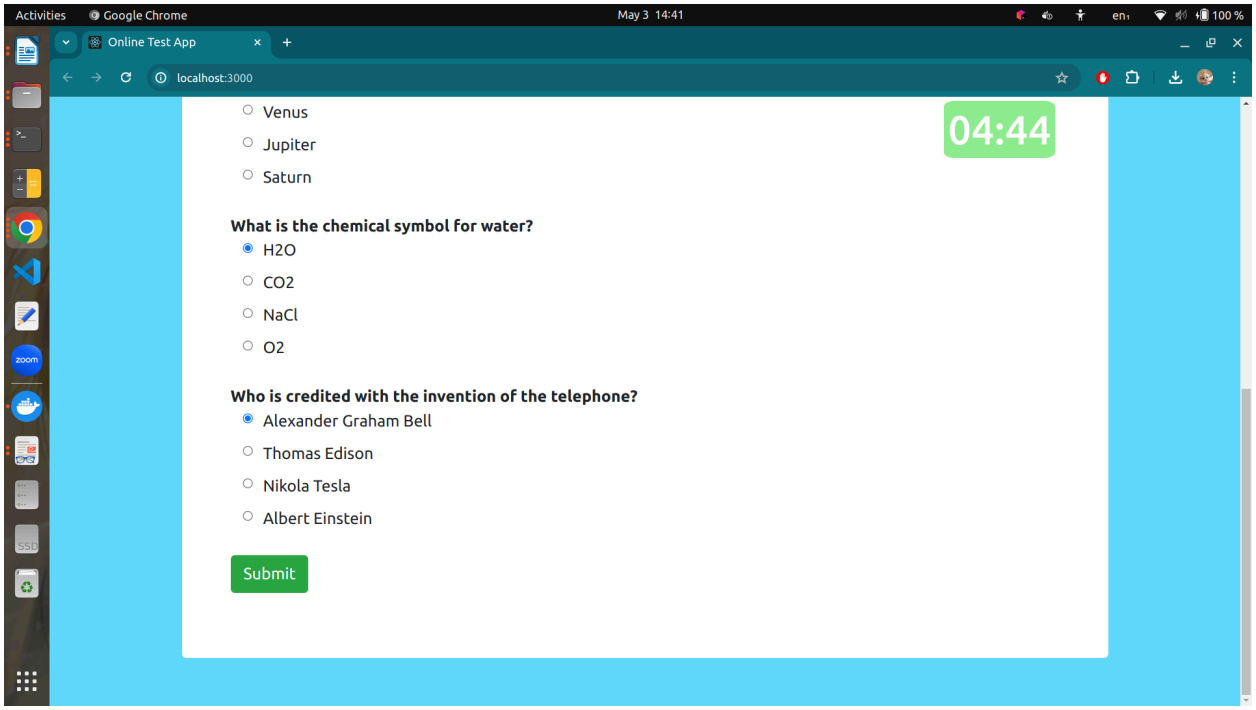
```
    return db;
  } catch (error) {
    console.error("Error connecting to MongoDB:", error);
    console.log('out with error');
    throw error;
  }

}

module.exports = { connect };
```

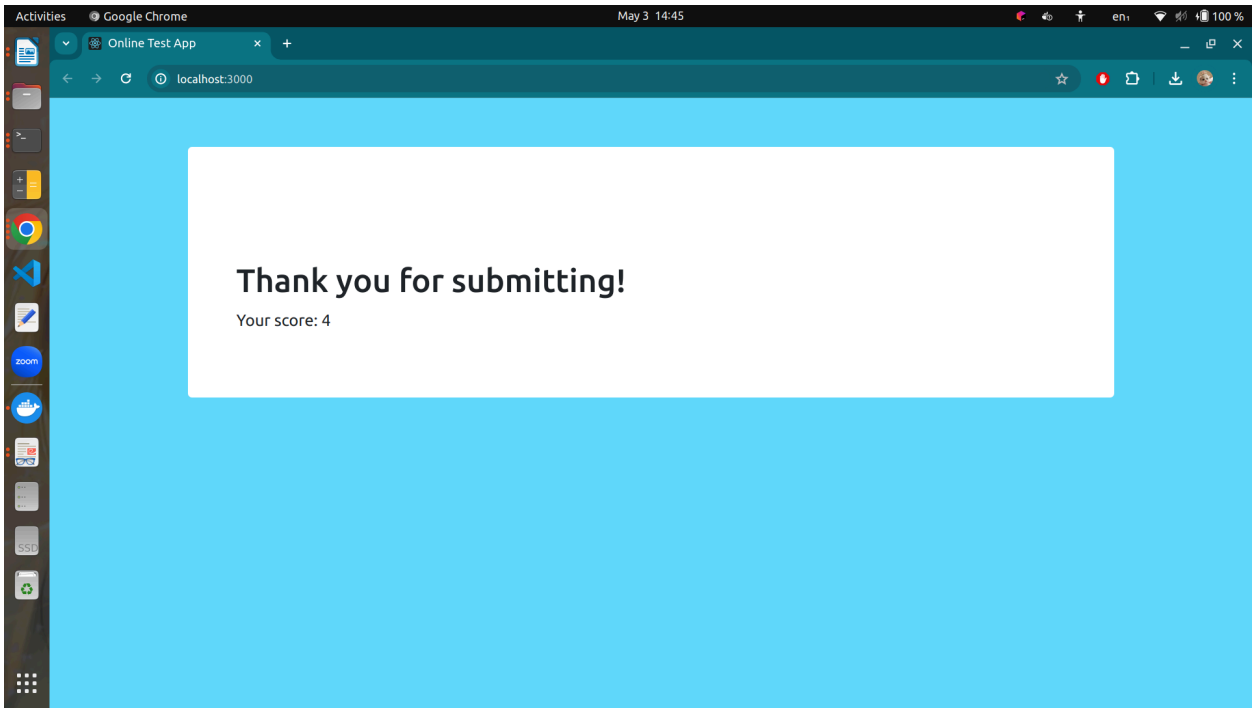
Output:

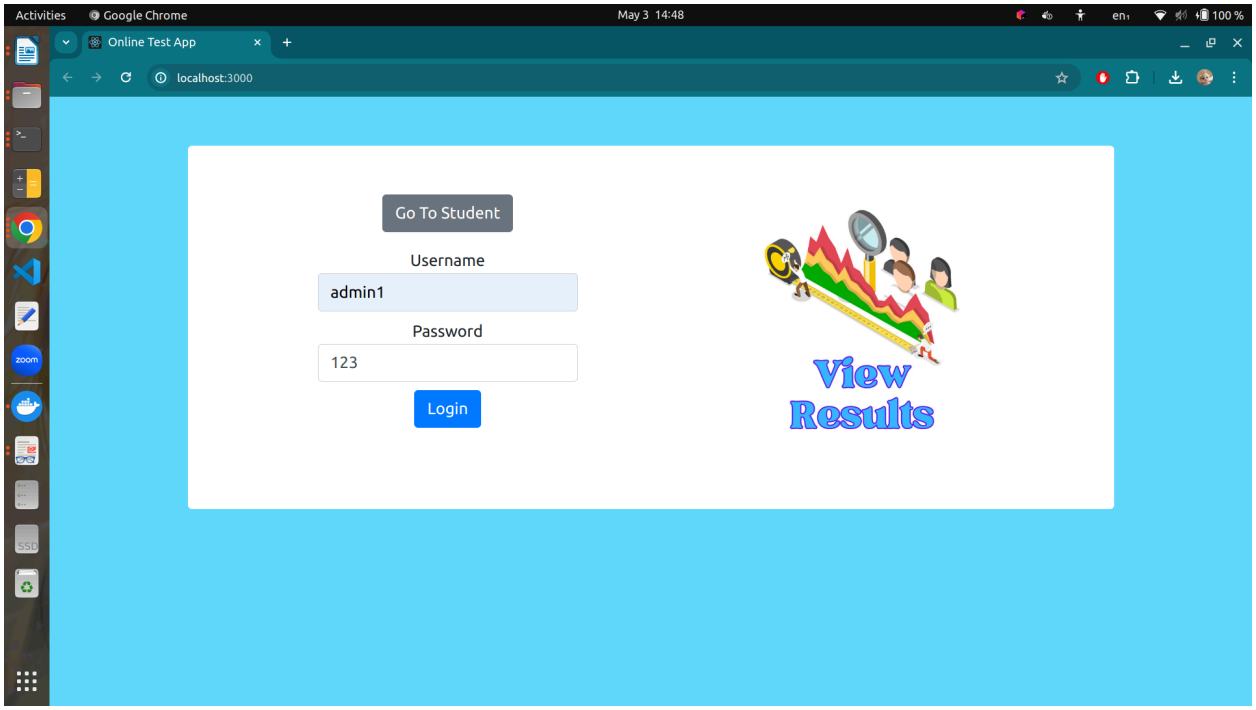


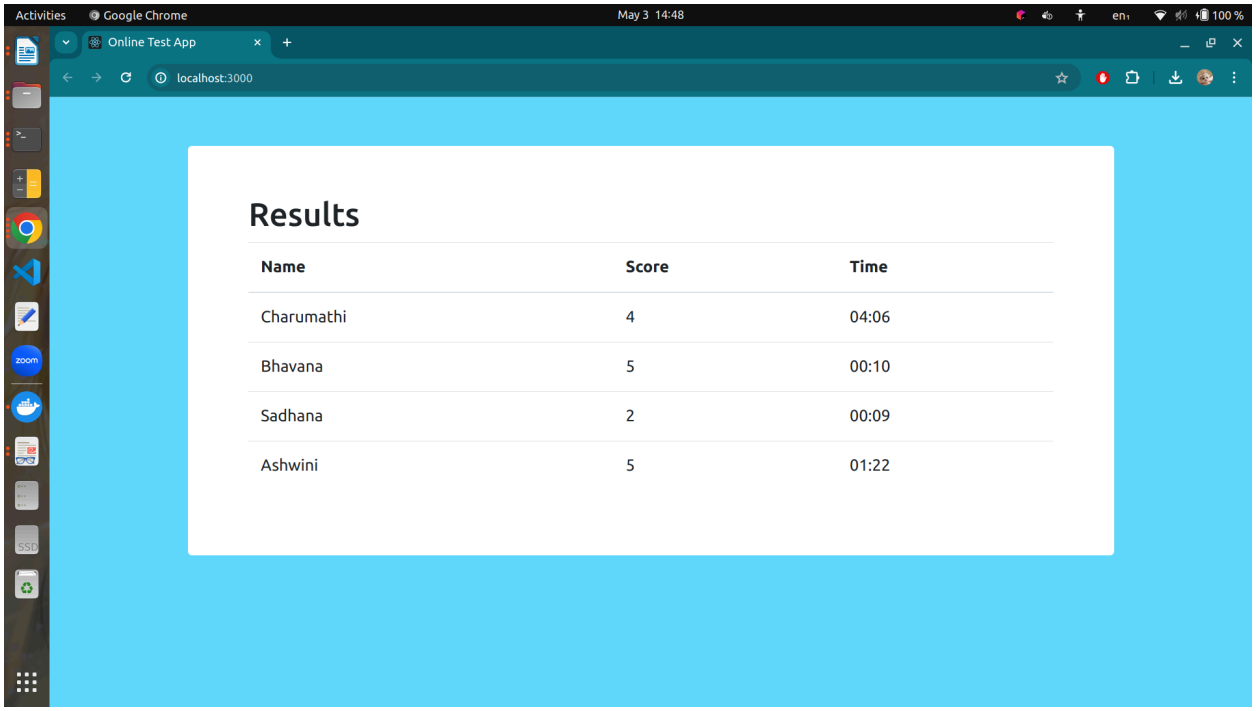

Activities
Google Chrome
May 3 14:40
en
100 %
Online Test App
localhost:3000
Go To Admin
Username
Charumathi
Password
234
Login
Start the test

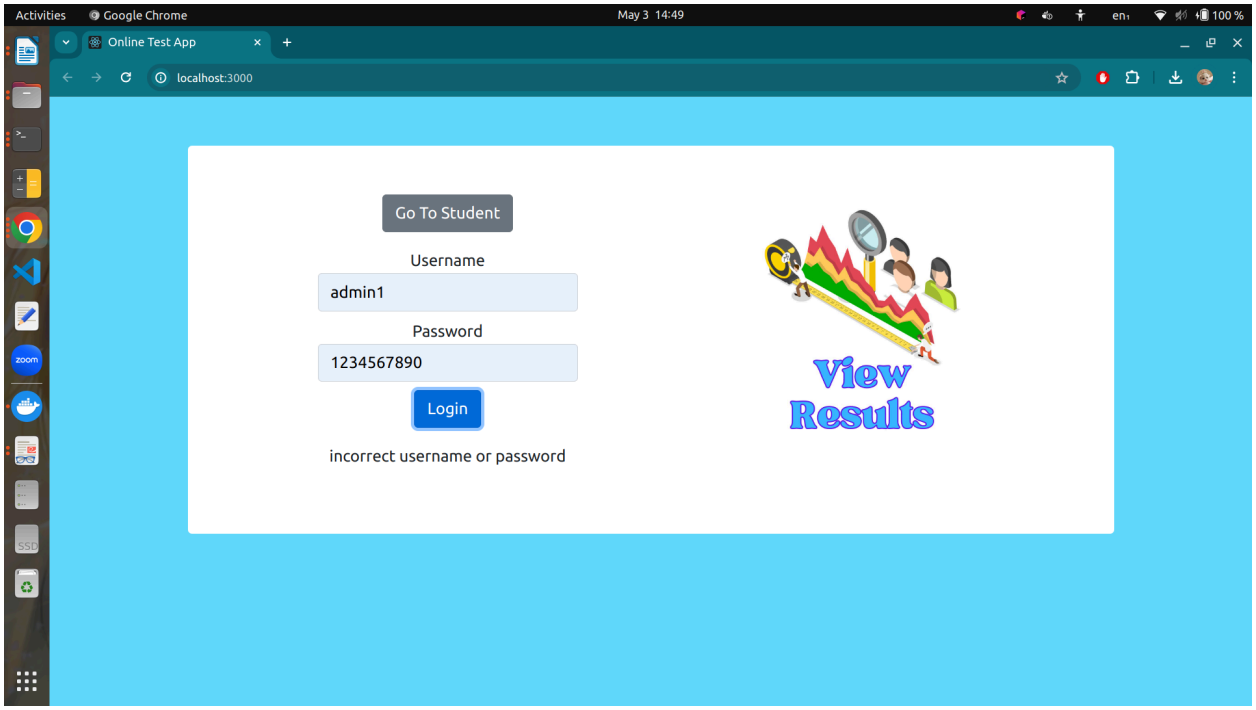

Activities
Google Chrome
May 3 14:41
en
100 %
Online Test App
localhost:3000
04:58
What is the capital of France?
Paris
Berlin
London
Rome
Who wrote 'To Kill a Mockingbird'?
Harper Lee
J.K. Rowling
Ernest Hemingway
Stephen King
Which planet is known as the Red Planet?
Mars
Venus
Jupiter


Activities
Google Chrome
May 3 14:41
Online Test App
localhost:3000
Venus
Jupiter
Saturn
04:44
What is the chemical symbol for water?
H2O
CO2
NaCl
O2
Who is credited with the invention of the telephone?
Alexander Graham Bell
Thomas Edison
Nikola Tesla
Albert Einstein
Submit


Activities
Google Chrome
May 3 14:45
Online Test App
localhost:3000
Thank you for submitting!
Your score: 4


Activities
Google Chrome
May 3 14:48
Online Test App
localhost:3000
Go To Student
Username
admin1
Password
123
Login
View Results


Activities
Google Chrome
May 3 14:48
Online Test App
localhost:3000
Results
Name
Score
Time
Charumathi
4
04:06
Bhavana
5
00:10
Sadhana
2
00:09
Ashwini
5
01:22



INFERENCE :

- Learnt to develop Quiz Application using MERN stack.
- Learnt to use MVC Architecture for better flow and management of the application.
- Learnt to use useEffect Hook for real-time application : timers.
- Learnt to create and alter MonogoDB collections.